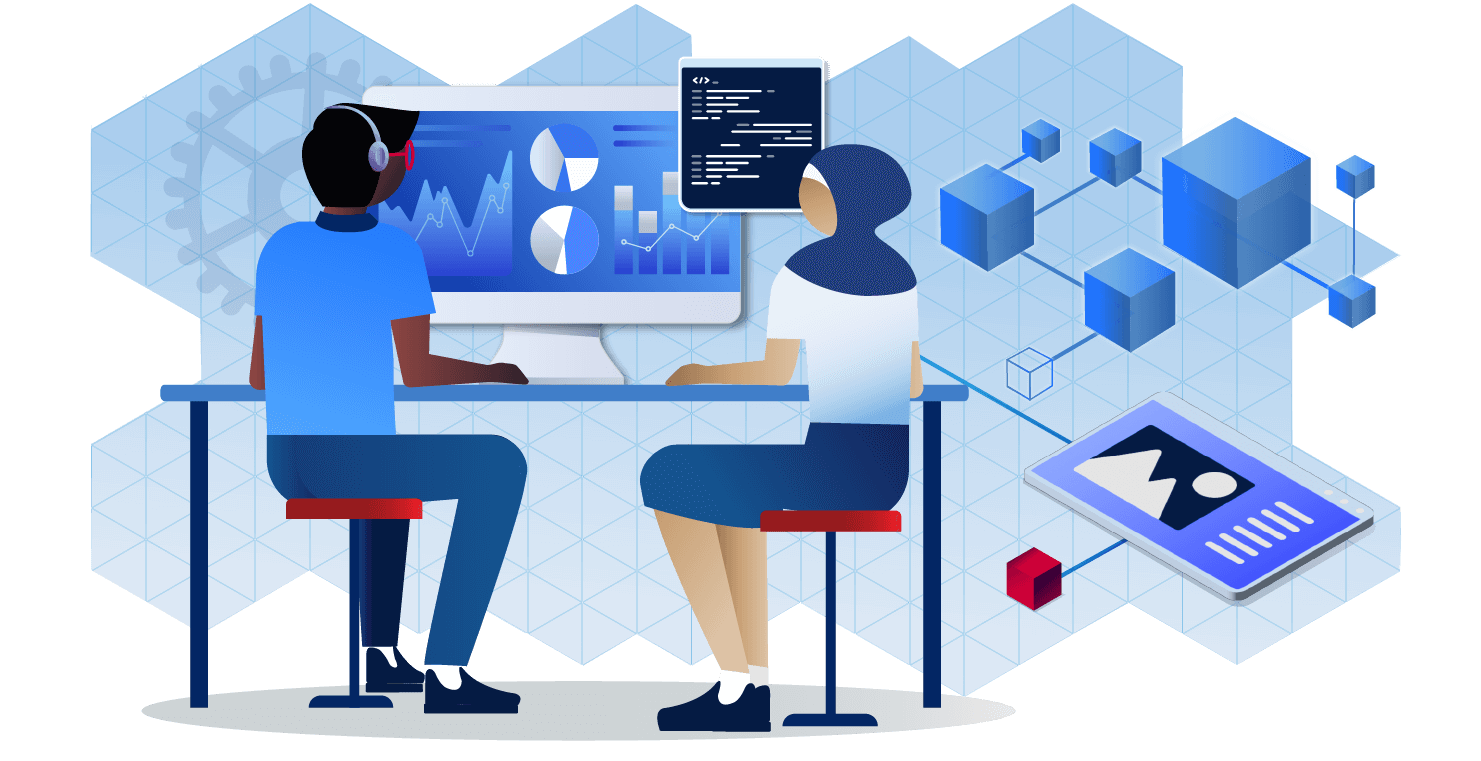
# Software Engineering Stage 6 (Year 12) – Sample Software Engineering Systems Report Template



Contents

[1. Identifying and defining 2](#_Toc196993426)

[1.1. Define and analyse problem requirements 2](#_Toc196993427)

[1.2. Tools to develop ideas and generate solutions 3](#_Toc196993428)

[2. Research and planning 4](#_Toc196993429)

[2.1. Project management 4](#_Toc196993430)

[2.2. Quality assurance 5](#_Toc196993431)

[2.3. Systems modelling 6](#_Toc196993432)

[3. Producing and implementing 12](#_Toc196993433)

[4. Testing and evaluating 13](#_Toc196993434)

[4.1. Evaluation of code 13](#_Toc196993435)

[4.2. Evaluation of solution 14](#_Toc196993436)

[Progress Diary 18](#_Toc196993437)

## 1. Identifying and defining

### 1.1. Define and analyse problem requirements

**Problem context**  
Students **analyse** the problem by **describing** each of its individual components and **explaining** how each of these components contribute to the problem needing resolution.

**Needs and opportunities**Students **describe** the needs of the new system to be built based on the problem context and using the table given below.

|  |  |
| --- | --- |
| Need | Description |
| 1. Being able to simulate an interaction between the client and our website/server | This is needed because it must demonstrate the basics of a digital store, the client will need to be able to interact with a clean UI and be able to see what is available cleanly and with smooth transitions. We need to get the basics of the website correct to produce and display a functional product |
| 2. Being able to store data in the back end of the website so we can logistically coordinate which orders go to which clients. | This need is about ensuring we have a database of client needs and our stock and cross referencing each other constantly to ensure client needs can be met to the highest of standards. |
| 3. It helps by saving my cousin time and stops him using his phone whilst driving (even phone holders cause distraction) | This website will help save him time on quotes and allow people to measure and request what type of Fly Screens they would need and is able to automate the process smoother with less hassle and headache. (automated quotes). It helps streamline the creation and design process and allows my cousin more time to build the Fly Screens and should be just easily |

**Boundaries**

Students **analyse** any limitations or boundaries in which this new system will need to operate. Boundaries can include but are not limited to: hardware, operating systems, security concerns etc*.*

The limitations and boundaries of this website will be quite simple, there is all the chance that most people don’t know how to give out relevant and necessary measurements which will result in possibly even more delays and lost product. We need to demonstrate to people how to take measurements and what material they would like exactly. The hardware that will limit us is because of a lack of equipment, NBN network delays due to an outdated network. This will limit us as it may delay the timeline to complete product significantly especially if outside factors such as the weather further disrupt the local satellite and internet connections. Security concerns are high, not only is there the potential for stolen banking information but also the potential that security threats could even delete or reform certain information which sends my cousin into multiple areas that didn’t want it. Deleting forms and orders, security concerns are grave. The Operating systems we chose are crucial.

The program must have:

A website which details the products available and the pricing options depending on client’s specific needs.  
A back-end database that allows for the storage of quotes and client’s orders stored neatly.  
My cousins phone number, so that clients can still contact him and ask for a quote and estimate how much it may cost.

The program should have:

A list of certain materials which would be suitable for your fly screen (customization), being able to detail your fly screen.

The possibility of making multiple orders for different places, doing it online rather than one massive quote.

The program could have:

Automated quoting system, given customization decent specifics and

### 1.2. Tools to develop ideas and generate solutions

**Identification of appropriate software development tools**Students **identify** appropriate tools for each of the given software development situations. They then **explain** how each tool is applicable or not to the current project.

|  |  |  |
| --- | --- | --- |
| Situation | Tool applicability | Reason |
| Brainstorming, mind-mapping and storyboards | Yes | Absolutely, this is applicable. It’s the basis of constructing a stable program and map out how it functions and demonstrates the potential for application in the real-world environment. |
| Data dictionaries | Yes |  |
| Algorithm design | Y / N |  |
| Code generation | Y / N |  |
| Testing and debugging | Y / N |  |
| Installation | Y / N |  |
| Maintenance | Yes |  |

**Implementation method**  
Students **explain** the applicability of the implementation method for the current project. These are normally: direct, phased, parallel and pilot.

I think that the implementation method that we should conduct would be a pilot testing, maybe with some exclusive clients due to the original system being in place. We need to know how this new website functions and gather feedback and then we can move into a phased or direct implementation method.

## 2. Research and planning

### 2.1. Project management

**Software development approach**Students **explain** the software development approach most applicable for this current project. These are normally: Waterfall, Agile and WAgile.

**Scheduling and task allocation**Students **develop** a Gantt Chart that details the tasks required to be completed, person or people assigned to each task, timeline that does not exceed the project due date, resources required. In addition, students **identify** any collaborative tools used. For example Repl.it, GitHub and so on.

### 2.2. Quality assurance

**Quality criteria**Students **explain** quality criteria based upon the needs from Section 1.1. These quality criteria should contain qualities, characteristics or components that need to be included or visible – based on Section 1.1. – by the end of the current project.

|  |  |
| --- | --- |
| Quality criteria | Explanation |
|  |  |
|  |  |
|  |  |

**Compliance and legislative requirements**Students **explain** compliance and legislative requirements their projects need to meet and how they plan to mitigate them where possible. For example, projects that deal with sensitive personal data being publicly available may fall under the Australian [NSW Privacy and Personal Information Act (1998)](https://legislation.nsw.gov.au/view/whole/html/inforce/current/act-1998-133#statusinformation) and/or [Federal Privacy Act (1988)](https://www.legislation.gov.au/Series/C2004A03712). Alternatively, international standards on information security management such as [ISO/IEC 27001](https://www.iso.org/standard/27001) may also be applicable.

|  |  |
| --- | --- |
| Compliance or legislative issue | Methods for mitigation |
|  |  |
|  |  |
|  |  |

### 2.3. Systems modelling

Students are to **develop** the given tables and diagrams. Students should consult the [Software Engineering Course Specifications](https://library.curriculum.nsw.edu.au/341419dc-8ec2-0289-7225-6db7f2d751ef/94e1eb0a-0df7-4dbe-9b72-5d5e0d17143a/software-engineering-11-12-higher-school-certificate-course-specifications.PDF) guide should they require further detail, exemplars or information. Each subsection below should be completed with Section 1.1. in mind.

**Data dictionaries and data types**Students take the needs identified in Section 1.1. of this Systems Report. For each need, students **identify** the variables required, data types, format for display, and so on.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Need |  |  |  |  |  |  |  |
| 1. | | | | | | | |
| Variable | **Data type** | **Format for display** | **Size in bytes** | **Size for display** | **Description** | **Example** | **Validation** |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |

| Need |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 2. | | | | | | | |
| Variable | **Data type** | **Format for display** | **Size in bytes** | **Size for display** | **Description** | **Example** | **Validation** |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |

| Need |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 3. | | | | | | | |
| Variable | **Data type** | **Format for display** | **Size in bytes** | **Size for display** | **Description** | **Example** | **Validation** |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |

**Data flow diagrams**

Students **develop** data flow diagrams (DFDs) at Level 0 and Level 1. These diagrams should explicitly include the variables from the data dictionaries previously identified as well as the needs identified in Section 1.1.

*Level 0*

*Level 1*

**Structure charts**Students **develop** structure charts demonstrating how the procedures, modules or components of the final solution are interconnected.

**Class diagrams**Students **develop** class diagrams demonstrating how each class is related to the other.

**Storyboards**Students **develop** storyboards, visually representing the software solutions they will build.

**Decision trees**Students **develop** decision trees to visually outline the logic flow and chain of decisions or selections the final solution will need.

**Algorithm design**Students **develop** algorithms using methods such as pseudocode or flowcharts to solve the problem and meet the needs from Section 1.1. These algorithms should explicitly include the variables from the data dictionaries created in the previous section.

## 3. Producing and implementing

**Solution to software problem**Students are to **include** screen shots of their final developed solution here. Each screenshot should include a caption that **explains** how it links to the:

* Needs identified in Section 1.1.
* Components of Section 2.3. such as the storyboards, data dictionaries and so on.

**Version control**Students **describe** what version control system or protocol was implemented.

## 4. Testing and evaluating

### 4.1. Evaluation of code

**Methodology to test and evaluate code**Students **explain** the methodologies used to test and evaluate code. Methodologies include:

* Unit, subsystem and system testing
* Black, white and grey box testing
* Quality assurance.

**Code optimisation**Students **explain** the methodologies used to optimise code so that it runs faster and more efficiently. Methodologies include:

* Dead code elimination
* Code movement
* Strength reduction
* Common sub-expression elimination
* Compile time evaluation – constant folding and constant propagation.

### 4.2. Evaluation of solution

**Analysis of feedback**Students **analyse** feedback given to them on the new system they have just created. This feedback can be in the form of an interview, survey, focus group, observation or any other applicable method. Students should also include overall positive, negative or neutral sentiments towards the new system in their response.

**Testing methods**Students **identify** the method or methods of testing used in this current project. For each they use, students are to **explain** how and why it was used.

|  |  |  |
| --- | --- | --- |
| Method | Applicability | Reasoning |
| Functional testing |  |  |
| Acceptance testing |  |  |
| Live data |  |  |
| Simulated data |  |  |
| Beta testing |  |  |
| Volume testing |  |  |

**Test data tables**Students **identify** variables which were used for either path and/or boundary testing. Students **develop** these test data tables based on their algorithms versus their real code. Students then **state** the reason for including said variables.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Variable | Maximum | Minimum | Default Value | Expected Output | Actual Output | Reason for Inclusion |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |

Boundary testing

Path testing

**Analysis of solution against quality success criteria**Students are to take each quality success criteria from Section 2.2 and place it here. For each quality criteria, **analyse** the components of the solution that met or did not meet each quality criteria. Give reasons why each success criteria were or were not met.

|  |  |  |
| --- | --- | --- |
| Quality criteria | Met? | Analysis |
|  |  |  |
|  |  |  |
|  |  |  |

## Progress Diary

30th April -